UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/799,904 | 03/12/2004 | Michael T. Kisamore | 018360/274115 | 8308 |

826 7590 02/05/2007

ALSTON & BIRD LLP
BANK OF AMERICA PLAZA
101 SOUTH TRYON STREET, SUITE 4000
CHARLOTTE, NC 28280-4000

| EXAMINER | |
|---|---|
| TAKELE, MESEKER | |
| ART UNIT | PAPER NUMBER |
| 2109 | |

| SHORTENED STATUTORY PERIOD OF RESPONSE | MAIL DATE | DELIVERY MODE |
|---|---|---|
| 3 MONTHS | 02/05/2007 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/799,904 | KISAMORE ET AL. | |
| | Examiner | Art Unit | |
| | Meseker Takele | 2109 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *05/12/2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-23* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *05/12/2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *5/23/06,7/19/06,6/7/04*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

### *Claim Rejections - 35 USC § 102*

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claim 1-9, 14, 15, 18 are rejected under 35 U.S.C. 102(b) as being anticipated by Halviatti et al. (US Patent No. 5475,843).

3. As to claim 1, Halviatti discloses a system for testing an application running on a target device (example, software testing automation system, application specific testing, target application, column 21, lines 37-47, column 2, line 37 and see figure 6) the system comprising a target device storing and executing a software test agent (example, storing, execution, Application Translation Units and Message Engine, see column 5, lines, 41-49, column 22, lines 36-41 and figure 3) wherein said application under test is also stored and executed on said target device (example, application under test, stored, column 21, lines 47-53, column 3, line, 4 and figure 6) a test development computer storing (example, mass storing, see column, 5, line 30) and executing a software test tool for testing and validating (example, test suites, input, output, see column 1, lines, 36-40) said application's rendering of output to a Graphical User Interface (GUI)(example, QA engineer see column 21, lines 15-21)
said test tool configured to communicate with said agent on said target device

(example, Model Manager, see figure 6 and column 22, lines 35-41) said software test tool executing a test script for testing and validating one or more aspects of said application's rendering of output to said GUI (example, user interface, input, output, see column 1, lines, 40-45) said test tool operable to generate requests to said agent to obtain information from and send events to controls associated with said GUI (see column 22, lines 23-50, column 27, lines 50-57) and said software test tool comprising a trap manager for handling trap events that can occur on said target device during execution of said test script (example, handlers see Column 9, line 59, column 10, line 6) said trap manager executed by said test development computer to allow a user to define a trap event and create an associated user-defined function for handling the trap event (example, trapped see column 12, line 57) automatically check (example, automatically controlled, see column 20, line 55) whether the trap event has occurred during execution of said test script; and upon detecting an occurrence of the trap event, execute said user-defined function to handle said trap event (example, handlers, see column 9, line 60, column 10, line 5).

As to claim 2, Halviatti discloses said trap event is a window that appears during execution of said test script, said window being defined by the user according to one or more window properties that can be checked by the agent to determine whether said window is actually the trap event (example event handlers, see column 7, lines 22-23).

As to claim 3, Halviatti does not discloses a capability to turn on and off during the execution of said test script, the checking of a trap event (it will be inherent that

there would be turn on and off during the execution of a test script)(see column 37, lines 38-41 and abstract).

As to claim 4,Halviatti does not discloses a capability to turn on and off during the execution of said test script, the checking of all trap events (it will be inherent that there would be turn on and off during the execution of a test script)(see abstract).

As to claim 5, Halviatti discloses a capability that allows a user to assign an input-event name to a grouping of multiple key sequences, said grouping of multiple keys sequences representing an input event that occurs on said target device during execution of said test script, wherein said input-event name can be used in place of said grouping each time said grouping is to be written into said test script (example, hot keys, see column 6, lines 30-33).

As to claim 6, Halviatti discloses a method of handling a trap event during execution of an automated test of an application running on a target device, the method comprising: creating a user-defined function for handling an occurrence of the trap event; defining the trap event using a software test tool being executed by a test development computer; checking to see whether the trap event has occurred on said target device during execution of said automated test; and upon detecting an occurrence of the trap event, executing said user-defined function to handle the trap event (example, trap event, see figure 4).

As to claim 7, Halviatti discloses, wherein said trap event is a window that appears during execution of said automated test, said window being defined according

to one or more window properties that can be checked to determine whether said window is actually the trap event (see column 41, lines 33-34).

As to claim 8, Halviatti discloses a method of handling a trap event during execution of an automated test of an application running on a target device the method comprising: creating a user-defined function for handling an occurrence of the trap event (example, handlers, see column 9, line 59 and column 10, line 6) defining the trap event using a software test tool executing on a test development computer; storing trap event data on the target device, said data capable of being used by an agent executing on the target device to detect the occurrence of the trap event (see column 22, lines 35-41, column 8, lines 47-58) monitoring with the agent to determine whether the trap event has occurred on the target device (example, monitors, see column 22, lines 35-41) upon detecting an occurrence of the trap event, transmitting a notification of the occurrence of the trap event from the agent to the test tool (see column 22, lines 23-50, column 27, lines 50-57) the notification comprising a user-defined name for uniquely referring to the trap event(see column 6, lines 30-33) accessing a table on the development computer using said trap event name to obtain a pointer (example, pointer, column 6, line 22) that points to the user-defined function for handling the trap event; and executing the user-defined function to handle the trap event (see column 6, 35-44).

As to claim 9, Halviatti discloses a system for testing an application running on a target device (see figure 6, and column 21, lines 37-47) the system comprising: a target device storing and executing a software test agent (example, Application Translation

Units and Message Engine, see figure 3 and column 22, lines 36-41) wherein said application under test is also stored and executed on said target device (example, Application Translation Units and Message Engine, see figure 3 and column 22, lines 36-41) a test development computer storing and executing a software test tool for testing and validating said application's rendering of output to a Graphical User Interface (GUI) )(example, QA engineer see column 21, lines 15-21) said test tool configured to communicate with said agent on said target device (example, Model Manager, see column 22, lines 35-41 and figure 6) said software test tool executing a test script for testing and validating one or more aspects of said application's rendering of output to said GUI, said test tool operable to generate requests to said agent to obtain information from and send events to controls associated with said GUI (see column 22, lines 23-50, column 27, lines 50-57) and said software test tool comprising a configuration manager for handling testing of said application ( example, model manager, see column 36, line 36) against multiple languages (example, different language, see column 30, lines 52-53) and platform configurations(example, different platform, Column 5, line, 20).

As to claim 14, Halviatti discloses a capability that allows a user to assign an input-event name to a grouping of multiple key sequences, said grouping of multiple key sequences representing an input event that occurs on said target device during execution of said test script, wherein said input-event name can be used in place of said grouping each time said grouping is to be written into said test script (example, hot keys, see column 6, lines 30-33).

As to claim 15 Halviatti discloses storing a configuration table having a plurality of user-defined configurations each configuration comprising a collection of value sets, each said value set comprising a collection of one or more related configuration items and corresponding values for said related items writing a test script that executes differently based on which user-defined configuration is loaded from said table (see column 42, lines 44-46); loading a user-defined configuration from said configuration table prior to execution of said test script; and executing said test script (see column 20, lines 32-50, column 37, lines 5-11).

As to claim 18, Halviatti discloses, a target device storing and executing a software test agent (example, Application Translation Units and Message Engine, see figure 3 and column 22, lines 36-41) wherein said application under test is also stored and executed on said target device (see figure 6 and column 21, lines 47-53) a test development computer storing and executing a software test tool for testing and validating said application's rendering of output to a Graphical User Interface (GUI)(example, QA engineer see column 21, lines 15-21) said test tool configured to communicate with said agent on said target device (example, Model Manager, see figure 6 and column 22, lines 35-41) said software test tool executing a test script for testing and validating one or more aspects of said application's rendering of output to said GUI ( see column 22, lines 23-50, column 27, lines 50-57) said test tool operable to generate requests to said agent to obtain information from and send events to controls associated with said GUI( see column 22, lines 23-50, column 27, lines 50-57) said software test tool comprising a configuration manager for handling testing of said

application against multiple languages (See column 26, line 20)and platform configurations(see column 5, line 13) and said software test tool comprising a trap manager for handling trap events that can occur on said target device during execution of said test script (example, trapped see column 12, lines 55-57) said trap manager executed by said test development computer to: allow a user to define a trap event and create an associated user-defined function for handling the trap event (example, trapped see column 12, lines 55-57) automatically check whether the trap event has occurred during execution of said test script; and upon detecting the occurrence of the trap event, execute said user-defined function to handle said trap event (see column 9, line 59, column 10, line6).

***Claim Rejections - 35 USC § 103***

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was

not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

5. Claim 10-13,16, 17, 19-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Halviatti et al. (US Patent number 5,475,843) in view of Janniro et al. (US Patent number 5,634,098).

As to claim 10, it is noted that Halviatti does not disclose a configuration table that includes one or more configurations, each said configuration comprising a collection of value sets, each said value set comprising a collection of one or more related configuration items and corresponding values for said related items. Janniro, from the same field of endeavor disclose, a configuration manager comprises a configuration table that includes one or more configurations, (example, plurality of environment configuration, see column 2, line, 43) each said configuration comprising a collection of value sets, each said value set comprising a collection of one or more related configuration items and corresponding values for said related items (example, configuration files, that specify value, see column 2 lines 6-18 and figure 4). It would have been obvious to one ordinary skill in the art to modify Halviatti with the features of configuration table that includes one or more configurations each said configuration comprising a collection of value sets, each said value set comprising a collection of one or more related configuration items and corresponding values for said related items as presented by Janniro. The motivation to combine the two references will help to

perform automated software testing based on values of environment variable specified in files stored in a hierarchical directory structure.

As to claim 11, Janniro discloses, configuration table is stored in a spreadsheet format for ease of editing by a user (example, directory 502, 504, 506, 508, 510, 512, 514, 515, and 518 see figure 5).

As to claim 12, Janniro discloses responsive to a user changing from a first value set to a second value set during execution of a test script, the test tool is configured to automatically delete all configuration item values associated with the first value set and reload said configuration items with corresponding values associated with the second value set (see column 10, lines 16-31).

As to claim 13, Janniro discloses, a capability that allows a user to get the value of a particular configuration item, and a capability that allows a user to set the value of a particular configuration item, during execution of a test script (see column 6, lines 20-22 and lines 58-67).

As to claim 16, Janniro discloses, the configuration table is stored in a spreadsheet format for ease of editing by a user (example, directory 502, 504, 506, 508, 510, 512, 514, 515, and 518 see figure 5).

As to claim 17, Janniro discloses, changing from a first value set to a second value set during execution of said test script; and responsive to said changing: automatically deleting all configuration item values associated with the first value set; and automatically reloading said configuration items with corresponding values associated with the second value set (see column 10, lines 16-31).

As to claim 19, Halviatti disclose, a method performed by a test development computer storing and executing a test tool operable to communicate with a target device storing and executing a test agent (example, computer-aided software testing system, GUI, see column, 1, lines 22-25) to interrogate a particular Graphical User Interface (GUI) control that is associated with an application under test running on said target device, the method comprising: storing a first table (example, control, target application, see column 18, lines 7-8) that comprises a list of entries, each said entry containing information corresponding to a particular GUI control associated with said application under test, the information associated with each said control a user-defined name for uniquely referring to that control (example, unique identifier, particular control, see column 33, lines 1-9 and figure 9c) a plurality of properties which define that control including a class name ( example, class name, column, 10 line, 61) that indicates a class type (example, type, column 31, line 4) to which that control belongs (example, particular control among its siblings, see column 33, lines 1-9 and figure 9) and a data field for optionally storing (see column 9, line 49) a control-specific identify flag (see column 29, lines 4-6), said control-specific identify flag (example, flags of the control, Column 16, lines 24, figure 5A) used for indicating which of said properties are to be used in identifying that control on said target device. Halviatti further discloses a class name for uniquely referring to that class of controls and a default identify flag for indicating which of said properties are to be used for identifying any control of that class type that does not have a control-specific identify flag associated with it, as defined in said first table (see figure 9C). It is noted that Halviatti does not specifically disclose

storing a second table that comprises a list of entries each said entry containing information related to a particular class of GUI controls associated with said application under test, the information associated with each said class of controls comprising: a class name for uniquely referring to that class of controls and a default identify flag for indicating which of said properties are to be used for identifying any control of that class type that does not have a control-specific identify flag associated with it, as defined in said first table. Janniro, from the same field of endeavor disclose, storing a second table that comprises a list of entries each said entry containing (see abstract) information related to a particular class of GUI controls (see column 5, line 6-9) associated with said application under test (see column 13, line 51) the information associated with each said class of controls (see figure 1). It would have been obvious to one ordinary skill in the art to have modified Halviatti with the features of storing a second table that comprises a list of entries each said entry containing information related to a particular class of GUI controls associated with said application under test the information associated with each said class of controls as taught by Janniro. The motivation to combine the two references will help to list and store executable routines in table.

As to claim 20, Halviatti disclose checking to see whether the control-specific identify-flag data field associated with said control is empty loading (example, load, column 32, line 16 and figure 9C-D) the identify flag from said control-specific identify flag data field, if said data field is not empty; if the control-specific identify-flag field is empty, loading the default identify flag from the second table using the class name

associated with said control to be interrogated (example, unique ID, fetched, see column 32, line 6-9 and figure 9C-D) generating a request to the agent to locate a control on the target device that matches said control to be interrogated, the agent locating a matching control by searching for a control which has a set of properties indicated by the identify flag obtained in the steps above that match the corresponding set of properties for said control to be interrogated, the matching control having a handle for accessing said control associated therewith; and receiving said handle for said matching control from the agent (example, control, null, flag, see column 39, lines 5-50 and figure 9C-D).

As to claim 21, Halviatti disclose a data field for optionally storing (example, storing, see column 43, line 4) a control-specific verify flag (see figure 5A) said control-specific verify flag used for indicating which of said properties are to be used when verifying that an occurrence of that control on said target device matches an expected state; and wherein the entry associated with each said class of controls listed in said second table further comprises a default verify flag, said default verify flag used for indicating which of said properties are to be used for verifying any control of that class type that does not have a control-specific verify flag associated with it, as defined in said first table .

As to claim 22, Halviatti disclose checking to see whether the control-specific verify-flag data field associated with said control to be interrogated is empty; loading the verify flag from said control-specific verify flag data field, if said data field is not empty; if the control-specific verify-flag field is empty, loading the default verify flag from the

second table using the class name associated with said control to be interrogated (it is inherent that if the control-specific identify-flag data field associated with said control is empty, loading the default identify flag from the second table (see column 27, line 63) interpreting the verify flag obtained in the steps above to determine which properties are to be used in verifying that said matching control matches said control to be interrogated (see column 2, lines 37-47) generating a request to the agent to retrieve actual property values associated with said matching control, said values retrieved comprising values for at least those properties indicated in the verify flag obtained in the steps above; and comparing said actual property values associated with said matching control, for only those properties indicated in the verify flag obtained above, to the corresponding property values defined in said first table for said control to be interrogated ((see column 2, lines 49-54).

As to claim 23, Halviatti disclose upon detecting a mismatch in any of the properties compared, generating an entry in a log file detailing the mismatch (see column 42, lines 47-64).

***Conclusion***

6. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

The US PG Pub 2003/0052917 by Dubovsky et al. is cited to teach automated test engine for GUI applications.

The US Patent number 5,499,359 by Vijaykumar et al. is cited to teach methods for relational database management system.

The US Patent number 5,600,789 by Parker et al. is cited to teach automated GUI interface testing.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Meseker Takele whose telephone number is (571) 270-1653. The examiner can normally be reached on Monday - Friday 7:30AM- 5:00PM est.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Xiao Wo can be reached on (571) 272-2100. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MT

XIAO WU
SUPERVISORY PATENT EXAMINER